IN THE SPECIFICATION:

Please amend the specification as follows:

Please replace the paragraph at page 5, lines 1 to 12, with the following:

In a third aspect of the invention, the state of the active file system is described by a set of metafiles; in particular, a bitmap (henceforth the "active map") describes which blocks are free and which are in use by the active file system. The inode file describes which blocks are used by each file, including the metafiles. The inode file itself is described by a special root inode, also known as the "fsinfo block." This copy of the root inode becomes the root of the snapshot. The root inode captures all required states for creating the snapshot such as the location of all files and directories in the file system. During subsequent updates of the active file system, the system consults the bitmap included in the snapshot (the "snapmap") to determine whether a block is free for reuse or belongs to a snapshot. This mechanism allows the active file system to keep track of which blocks each snapshot uses without recording any additional bookkeeping information in the file system.

Please replace the paragraphs at page 8, line 16, to page 9, line 17, with the following:

- U.S. Patent Application Serial No. 09/642,063, Express Mail Mailing No. EL 524781089US, filed August 18, 2000, in the name of Blake LEWIS, attorney docket number 103.1033.01, titled "Reserving File System Blocks"

- U.S. Patent Application Serial No. 09/642,062, Express Mail Mailing No. EL524780242US, filed August 18, 2000, in the name of Rajesh SUNDARAM, attorney docket number 103.1034.01, titled "Dynamic Data Storage"

- U.S. Patent Application Serial No. 09/642,066, Express Mail Mailing No. EL524780256US, filed August 18, 2000, in the name of Ray CHEN, attorney docket number 103.1047.01, titled "manipulation of Zombie Files and Evil-Twin Files"

A2

- U.S. Patent Application Serial No. 09/642,064, in the names of Scott SCHOENTHAL, Express Mailing Number EL524781075US, titled "Persistent and Reliable Delivery of Event Messages", assigned to the same assignee, attorney docket number 103.1048.01, and all pending cases claiming the priority thereof.

and

- U.S. Patent Application Serial 09/642,065, in the names of Douglas P. DOUCETTE, Express Mailing Number EL524781092US, titled "Improved Space Allocation in a Write Anywhere File System", assigned to the same assignee, attorney docket number 103.1045.01, and all pending cases claiming the priority thereof.

Please replace the paragraph at page 15, line 17, to page 16, line 6, with the following:

The active map 115 of the active file system 110 is a bitmap associated with the vacancy of blocks for the active file system 110. The respective snapmaps 140, 145, 150 and 155 are active maps that can be associated with particular snapshots 120, 125, 130 and 135. A summary map 160 is an inclusive OR of the snapmaps 140, 145, 150 and 155. Also shown are other blocks 115 including double indirect blocks 130 and 132, indirect blocks 165, 166 and 167 and data blocks 170, 171, 172 and 173. Finally, Figure 1 shows the spacemap 180 including a collection of spacemap blocks of numbers 182, 184, 186, 188 and 190.

A3

Please replace the paragraph at page 17, line 17, to page 19, line 2, with the following four paragraphs:

Another inode block in the inode file 105 is inode block N 195. This block includes a set of pointers to a collection of snapshots 120, 125, 130 and 135 of the volume. Each snapshot includes all the information of a root block and is equivalent to an older root block from a previous active file system. The snapshot 120 may be created at any past CP. Regardless when the snapshot is created, the snapshot is an exact copy of the active file system at that time.

A4

The newest snapshot 120 includes a collection of pointers that are aimed directly or indirectly to the same inode file 105 as the root block 100 of the active file system 110. As the active file system 110 changes (generally from writing files, deleting files, changing attributes of files, renaming file, modifying their contents and related activities), the active file system and snapshot will diverge over time. Given the slow rate of divergence of an active file system from a snapshot, any two snapshots will share many of the same blocks.

The newest snapshot 120 is associated with snapmap 140. Snapmap 140 is a bit map that is initially identical to the active map 115. The older snapshots 125, 130 and 135 have a corresponding collection of snapmaps 145, 150 and 155. Like the active map 115, these snapmaps 145, 150 and 155 include a set of blocks including bitmaps that correspond to allocated and free blocks for the particular CP when the particular snapmaps 145, 150 and 155 were created.

A4

Any active file system may have a structure that includes pointers to one or more snapshots. Snapshots are identical to the active file system when they are created. It follows that snapshots contain pointers to older snapshots. There can be a large number of previous snapshots in any active file system or snapshot. In the event that there are no snapshot, there will be no pointers in the active file system.

Please replace the paragraph at page 20, lines 1 to 8, with the following:

A summary map 160 is created by using an IOR (inclusive OR) operation 139 on the snapmaps 140, 145, 150 and 155. Like the active map 115 and the snapmaps 140, 145, 150 and 155, the summary map 160 is a file whose data blocks (1, 2, 3, ...Q) contained a bit map. Each bit in each block of the summary map describes the allocation status of one block in the system with "1" being allocated and "0" being free. The summary map 160 describes the allocated and free blocks of the entire volume from all the snapshots 120, 125, 130 and 135 combined. The use of the summary file 160 is to avoid overwriting blocks in use by snapshots.

A5

Please replace the paragraphs at page 20, line 10, to page 21, line 15, with the following:

An IOR operation on sets of blocks (such as 1,024 blocks) of the active map 115 and the summary map 160 produces a spacemap 180. Unlike the active map 115 and the summary map 160, which are a set of blocks containing bitmaps, the spacemap 180 is a set of blocks including 182, 184, 186, 188 and 190 containing arrays of binary numbers. The binary numbers in the array represent the addition of all the vacant blocks in a region containing a fixed number of blocks, such as 1,024 blocks. The array of binary numbers in the single spacemap block 181 represents the allocation of all blocks for all snapshots and the active file system in one range of 1,024 blocks. Each of the binary numbers 182, 184, 186, 188 and 190 in the array are a fixed length. In a preferred embodiment, the binary numbers are 16 bit numbers, although only 10 bits are used.

A6

In a preferred embodiment, the large spacemap array binary number 182 (0000001111111110=1,021 in decimal units) tells the file system that the corresponding range is relatively full. In such embodiments, the largest binary number 00001111111111 (1,023 in decimal) represents a range containing at most one empty block. The small binary number 184 (0000000000001110=13 in decimal units) instructs the file system that the related range is relatively empty. The spacemap 180 is thus a representation in a very compact form of the allocation of all the blocks in the volume broken into 1,024 block sections. Each 16 bit number in the array of the spacemap 180 corresponds to the allocations of blocks in the range containing 1,024 blocks or about 4 MB. Each spacemap block 180 has about 2,000 binary numbers in the array and they describe the allocation status for 8 GB. Unlike the summary map 120, the spacemap block 180 needs to be determined whenever a file needs to be written.

A6

Please replace the paragraphs at page 22, line 6, to page 23, line 14, with the following:

The snapmap 205 of the previous active file system, snapshot #1 201, is a bitmap associated with the vacancy of blocks for snapshot #1 201. The respective snapmaps 225, 230 and 235 are earlier active maps that can be associated with particular snapshots 210, 215 and 220. A summary map 245 is an inclusive OR of the snapmaps 225, 230 and 235. Also shown are other blocks 211 including double indirect blocks 240 and 241, indirect blocks 250, 251 and

A7

252, and data blocks 260, 261, 262, and 263. Finally, Figure 2 shows the spacemap 270 of snapshot #1 201 including a collection of spacemap blocks of binary numbers.

The old root block 200 includes a collection of pointers that were written to the previous active file system when the system had reached the previous CP. The pointers are aimed at a set of indirect (or triple indirect, or double indirect) inode blocks (not shown) or directly to the inode file 202 consisting of a set of blocks known as inode blocks 281, 282, 283, 284 and 285.

A7

An inode block 281 in the inode file 202 points to other blocks 328 in the old root block 200 starting with double indirect blocks 240 and 241 (there could also be triple indirect blocks). The double indirect blocks 240 and 241 include pointers to indirect blocks 250, 251 and 252. The indirect blocks 250, 251 and 252 include pointers that are directed to data leaf blocks 260, 261, 262, and 263 of the snapshot #1 201.

Inode block 283 in the inode file 202 points to a set of blocks (1, 2, 3, ..., P) called the snap map 205. Each block in the snap map 205 is a bitmap where each bit corresponds to a block in the entire volume. A "1" in a particular position in the bitmap correlates with a particular allocated block in the snapshot #1 201. Conversely, a "0" correlates to the particular block being free for allocation in the old root block 200. Each block in the snap map 205 can describe up to 32K blocks or 128 MB.

Please replace the paragraph at page 24, lines 3 to 6, with the following:

A8

Snapshot #1 201 also includes an old summary map 245 and old spacemap blocks 270. Although these blocks of data are included in snapshot #1 201 and previous snapshots, in a preferred embodiment, this data is not used by the active file system.

Please replace the paragraphs at page 24, line 13, to page 25, line 10, with the following:

A9

A method 300 is performed by the file system 110. Although the method 300 is described serially, the steps of the method 300 can be performed by separate elements in conjunction or in parallel, whether asynchronously, in a pipelined manner, or otherwise. There is no particular requirement that the method 300 be performed in the same order in which this description lists the steps, except where so indicated.

At a flow point 305, the file system 110 is ready to perform a method 300.

At a step 310, a user will request a snapshot of the file system 110.

At a step 315, a timer associated with the file system 110 initiates the creation of a new snapshot.

At a step 320, the file system 110 receives a request to make a snapshot.

At a step 325, the file system 110 creates a new file.

Please replace the paragraphs at page 25, line 15, to page 26, line 3, with the following:

At a step 335, the file system 110 makes the file read only.

A9

At a step 340, the file system 110 updates the new summary map by using an inclusive OR of the most recent snapmap and the existing summary file. This step must be done before any blocks are freed in the corresponding active map block. If multiple snapshots are created such that the processing overlaps in time, the update in step 340 need only be done for the most recently created snapshot.

Please amend the abstract to read as follows:

An improved method and apparatus for creating a snapshot of a file system. A record of which blocks are being used by a snapshot is included in the snapshot itself, allowing effectively instantaneous snapshot creation and deletion. The state of the active file system is described by a set of metafiles; in particular, a bitmap (henceforth the "active map") describes

A10

which blocks are free and which are in use. The inode file describes which blocks are used by each file, including the metafiles. The inode file itself is described by a special root inode, also known as the "fsinfo block". The system begins creating a new snapshot by making a copy of the root inode. This copy of the root inode becomes the root of the snapshot.

A10